基于动态 Bloomfilter 的云存储安全去重方案 *

王平雁，柳　毅

(广东工业大学 计算机学院，广州 510006)

摘　要：现有的所有权证明去重方案容易遭受诚实但好奇服务器的威胁影响，借助可信第三方解决该问题将导致开销过大。基于动态 Bloom filter 提出一种改进的、无需可信第三方的所有权证明安全去重方案，采用收敛加密算法抵抗诚实但好奇的服务器，并通过服务器检查数据块密文和标签的一致性来防止数据污染攻击。此外，采用密钥链机制对收敛密钥进行管理，解决了现有方案中收敛密钥占用过多存储空间的问题。分析与比较表明，该方案具有较小的密钥存储开销和传输开销。

关键词：云存储；数据去重；Bloom filter；收敛加密；所有权证明

中图分类号：TP309.2　　doi: 10.3969/j.issn.1001-3695.2018.07.0432

Secure deduplication scheme based on dynamic Bloom filter in cloud storage

Wang Pingyan, Liu Yi

(*School of Computers Guangdong University of Technology*, *Guangzhou* 510006, *China*)

Abstract: Existing Proof of Ownership schemes are vulnerable to the threat of honest-but-curious servers. With the help of a trusted third party to solve the problem can lead to high overhead. This paper proposed an improved Proof of Ownership scheme based on Dynamic Bloom Filter, which doesn't require a trusted third party. The scheme uses convergent encryption against honest-but-curious servers. To resist data poisoning attack, it checks whether the encrypted blocks correspond with the tokens at server side. Moreover, a key chaining mechanism is used to solve the problem that convergent keys require too much storage space in existing schemes. Analyses and comparisons show that the scheme has lower key storage overhead and transferring overhead.

Key words: cloud storage; data deduplication; bloom filter; convergent encryption; proof of ownership

0　引言

随着云计算技术的不断发展，越来越多的用户选择将数据外包给云端进行存储和管理。The CISCO Global Cloud Index(2015-2020)显示[1]，2015 年，全球数据中心业务总量为 4.7ZB，其中云数据业务 3.8ZB，占比 81%；预计在 2020 年，全球数据中心业务总量增长三倍将达到 15.3ZB，其中云数据业务 14.1ZB，占比 92%。面对空前庞大规模的数据量，如何经济、高效、安全地进行数据存储，是云服务提供商迫切需要解决的一个难题。

数据去重（deduplication）技术，也称为重复数据删除技术，是一种消除冗余文件或文件内部冗余数据块的技术。由于该技术仅保留一份数据副本，因此可以极大地节省云存储空间[2,3]。

典型的存储系统常常采取文件的摘要作为用户的文件拥有凭证，导致攻击者仅凭借文件的摘要就可以获得整个文件[4]。针对于此，Halevi 等人[5]首次提出了所有权证明（proof of ownership，PoW）的概念，用户必须通过所有权证明才能拥有文件的权限，但该方案客户端的计算开销和 I/O 开销过大。Di Pietro 等人[6]提出一种改进的方案 s-PoW，通过选取文件中随机的比特来对用户进行验证，该方案提高了客户端的效率和减小带宽消耗，但在服务器端效率较低。Blasco 等人[7]提出基于 Bloom filter 的方案 bf-PoW，云服务器通过 Bloom filter 能够快速验证用户的响应结果，该方案在服务器端相比 s-PoW 方案效率更高，然而存在着 Bloom filter 误判率随元素增加而不断增加的问题。Yu 等人[8]提出了一种适用于移动云计算的 PoW 去重方案，提高了客户端的效率。由于上述方案未考虑数据机密性，容易遭受诚实但好奇（honest-but-curious）服务器的威胁[9]，一些方案[10-14]借助可信第三方进行数据管理，从而保证了用户数据的机密性。然而，增加可信第三方使得开销过大，在实际的云服务系统中难以实现。González-Manzano 等人[15]提出一种结合收敛加密算法、无可信第三方的 ce-PoW 方案，有效防止诚实但好奇的服务器查看文件，同时还考虑了数据污染攻击的问

收稿日期：2018-07-13；**修回日期：**2018-08-31　**基金项目：**国家自然科学基金资助项目（61572144）

作者简介：王平雁（1990-），男，广东湛江人，硕士研究生，主要研究方向为云计算安全（1126877291@qq.com）；柳毅（1976-），男，教授，博士，主要研究方向为云计算机安全、网络安全等.

题，但该方案收敛密钥占用了客户端较大的存储空间。刘竹松等人[16]基于 Bloom filter 提出一种结合收敛加密的高效安全去重方案，但该方案未考虑 Bloom filter 误判率增长的问题，并且不能抵抗污染攻击。张曙光等人[17]提出一种无需可信第三方的自适应重复数据删除方案，首先检查数据的流行度，当数据为非流行数据时，借助 PAKE 协议进行密钥传递。由于 PAKE 协议共享密钥需要双方同时在线，因此该方案在实用性方面受到限制。

针对上述方案存在的缺陷，本文基于动态 Bloom filter 提出了一种无需可信第三方的所有权证明方案 DBF-Dedup。本方案通过对 Bloom Filter 的状态进行动态管理，有效缓解 Bloom Filter 误判率的快速增长问题；采用收敛加密算法保证了数据机密性，同时采用密钥链机制来管理收敛密钥，解决了密钥管理的难题。

1　问题描述

1.1　系统模型

如图 1 所示，本文方案的系统模型包括云服务提供商（CSP）和用户（User）。

a)云服务提供商（CSP）。提供外包数据服务的实体，由主服务器和存储服务器构成。主服务器负责文件重复性检测、生成挑战数据和验证标签等；存储服务器负责存储文件的数据块密文，向合法用户发送其申请访问的文件。

b)用户（user）。外包数据存储到云存储系统的实体。对于云存储系统中未存在的数据，用户只需要上传一次；对于云存储系统中已经存在的数据，用户只需要通过所有权证明即可获得数据权限，不需要重复上传，节省了带宽消耗。

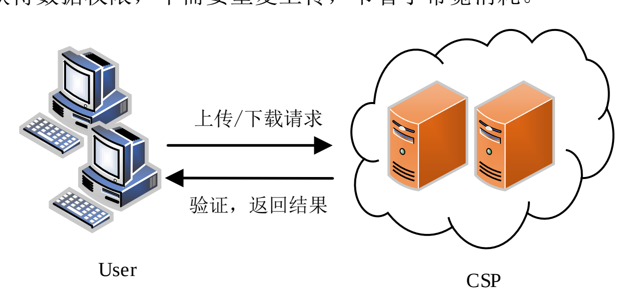


图 1　系统模型

1.2　威胁模型

在本文方案的威胁模型中，将攻击者来源分两大类：

a)内部攻击者。通常指的是来自 CSP 内部的恶意员工，他们的目标是从用户存储的数据中获取有价值的信息，甚至会与外部攻击者进行串谋。另外，内部攻击者可能为了达到某些目的而破坏数据的完整性。

b)外部攻击者。通常指的是来自 CSP 外部的攻击者，他们企图冒充合法的用户来获取云存储系统中的数据。外部攻击者可能从其他地方得到部分信息，比如文件的摘要，希望通过该信息来获得全部的数据。外部攻击者可能与合法的用户进行合作，或与内部攻击者进行串谋。

2　预备知识

2.1　Bloom filter

Bloom filter[18]是一种高效的概率性数据结构，用于判断某个元素是否属于特定的集合，通常由 1 个二进制向量和 k 个相互独立的哈希函数组成。设一个 Bloom filter 中有 m 比特的二进制向量，初始化所有比特位为 0。集合中有 n 个元素，每个元素通过计算 k 个哈希函数$\{H_1, H_2, \ldots, H_k\}$映射到$\{1, 2, \ldots, m\}$的范围中。当插入元素 x 时，将 k 个哈希函数映射的位置 $H_i(x)$ 置为 1。如图 2 所示，n=2，k=3，箭头指向的地方即是哈希函数映射的比特位，将其置为 1。

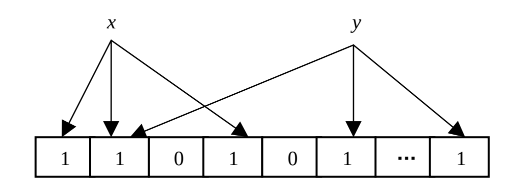


图 2　Bloom Filter 映射方法示意图

当查询某个数据对象 s 时，计算$\{H_1(s), H_2(s), \ldots, H_k(s)\}$，查看映射位置是否全为 1，若不全为 1，则判断该集合一定不包含 s；若全为 1，s 很可能属于该集合，但也有一定概率出现误判。假设某个不属于集合的元素的 k 个函数映射位置恰好全为 1，则会误判该元素属于集合，这种现象被称为假阳性（false positives），而且当集合中元素越多误判率就越高。元素存在的误判率为 p_f：

$$p_f = (1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-\frac{kn}{m}})^k \tag{1}$$

除此之外，Bloom Filter 另一个缺陷是不能删除已存在的元素。

针对以上缺点，采用动态 Bloom filter（dynamic Bloom filter, DBF）[19]能有效控制误判率的增加。动态 Bloom filter 由若干个标准 Bloom filter（standard Bloom filter,SBF）组成。初始阶段 DBF 中的 SBF 数量是 1，状态是 *active*，表示误判率小于上限值；随着不断插入新的元素，最后状态将变为 *full*，即误判率达到上限值，此时增加一个新的 SBF，确保状态一直是 *active*。通过监控 SBF 的状态并保持动态更新，有效控制了误判率。除了插入操作，DBF 还可以执行查询、删除、合并等操作。

DBF 需要初始化以下几个参数：DBF 的最大误判率，SBF 数量 s 的上限值，SBF 的最大误判率，集合中元素个数 n，单个 SBF 的大小 m，单个 SBF 的容量 c，单个 SBF 的哈希函数数量 k。易知 $s=\lceil n/c \rceil$。

2.2　收敛加密

收敛加密(convergent encryption,CE)算法[20]是一种确定性加密算法，收敛密钥由数据内容产生，确保由相同的数据得到相同的密钥。一个收敛加密方案由以下基本密码原语构成：

$KeyGen(F)$ => K：收敛密钥产生算法。输入数据明文 F，输出收敛密钥 K。其中，密钥 K 通常由哈希函数 H 生成，即

$K=H(F)$。

$Encrypt(K, F) => C$：加密算法。输入收敛密钥 K 和明文 F，输出经过加密后的密文 C。

$Decrypt(K, C) => F$：解密算法。输入收敛密钥 K 和密文 C，输出经过解密后的明文 F。

$TagGen(F) => T_F$：标签产生算法。输入数据明文 F，输出对应的标签 T_F。

3　方案设计

3.1　符号说明

k 是数据块划分参数，PRF 表示伪随机函数，*seed* 是初始化种子，B_i 表示第 i 个数据块，C_i 表示第 i 个密文数据块，K_i 表示第 i 个数据块的收敛密钥，C_{Ki} 示第 i 个数据块的收敛密钥密文，$token_i$ 表示第 i 个密文数据块的标签，h_c 是$\{token_i\}$的摘要，J 是挑战数据块的数量值。

3.2　系统初始化

a)用户（user）。初始化收敛加密方案的四个算法(*KeyGen*, *Encrypt*, *Decrypt*, *TagGen*)和用于所有权证明的 PoW 算法。

b)云服务提供商（CSP）。初始化用于存储元数据如文件标签、动态 Bloom Filter 的存储系统，包括 k 个哈希函数$\{H_1, H_2, \ldots, H_k\}$。

3.3　文件上传

文件上传的主要流程如图 3 所示。具体的步骤如下所述：

*a)*用户发送文件 F 的大小至服务器。

*b)*服务器返回数据块划分参数 k。

*c)*用户根据 k 将文件划分为 n 个数据块$\{B_i\}(1 \le i \le n)$，计算得收敛密钥 $K_i = KeyGen(B_i)$，利用收敛密钥计算密文数据块 $C_i = Encrypt(K_i, B_i)$，计算密文数据块的标签 $token_i = H(B_i)$，计算所有标签的摘要 $h_c = H(token)$。用户上传 h_c 至服务器。

*d)*服务器检查是否已存在 h_c，若存在重复，则转到 *Step*5，对用户发起验证挑战；否则转到 *Step*6，要求用户上传文件。

e)（存在重复）。（a）服务器发起挑战，随机选择 J 个数据块索引发送给用户；（b）用户响应挑战，将对应的 J 个数据块的标签$\{token_j\}$返回服务器；（c）服务器将 *token* 值作为 *seed* 初始化 PRF 得到 J 个输出结果，利用 Bloom Filter 哈希函数$\{H_1, H_2, \ldots, H_k\}$对输出结果计算后映射到比特位，若结果全为 1，则表示用户通过文件所有权证明，服务器记录用户获得文件权限；否则，用户挑战失败。（d）用户获得文件权限后，将第一个数据块的收敛密钥 K_1 保存在本地，用于以后从 CSP 中下载文件时对密钥密文进行解密。

f)（无重复）。（a）用户采用密钥链机制 $C_{Ki} = E(K_{i-1}, K_i)(2 \le i \le n)$计算得到收敛密钥的密文$\{C_{Ki}\}$，即利用前一个密钥加密后一个密钥;(b)用户保存 K_1 在本地，将$\{C_i\}$、$\{C_{Ki}\}$上传到服务器；(c)为了防止污染攻击，服务器根据$\{C_i\}$重新计算$\{token_i\}$和 h'_c，与用户上传的 h_c 进行对比，若 $h'_c = h_c$ 成立，则数据上传成功；否则，说明密文数据块与标签不一致，对该用户返回警告，数据上传失败；（d）数据上传成功后，服务器创建一个动态 Bloom filter，服务器将 *token* 值作为 *seed* 初始化 PRF 并产生相应的输出结果，将输出结果插入到 Bloom filter，对应的比特位都置为 1；（e）检测 Bloom filter 的状态（*active* 或 *full*），若为 *full*，则创建一个新的 Bloom filter。

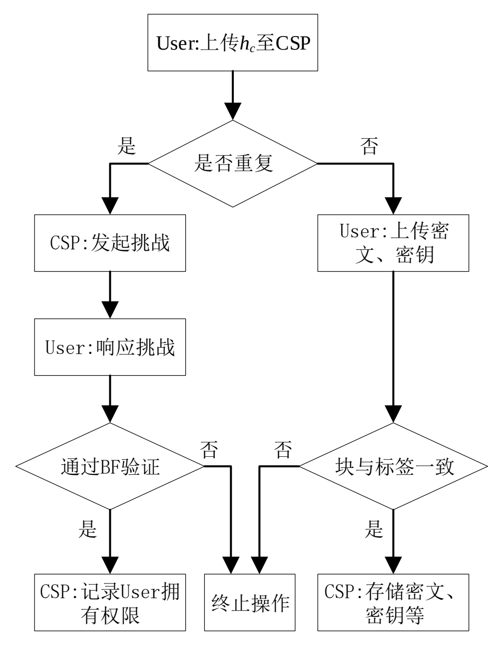


图 3　文件上传流程图

3.4　文件下载

用户请求下载文件 F 时的步骤如下：

*a)*用户向服务器请求下载文件 F。

*b)*服务器收到请求，首先核对用户的 ID：若无访问权限则返回警告；若有访问权限则将文件密文块$\{C_i\}$和收敛密钥的密文$\{C_{Ki}\}$返回给用户。

*c)*用户利用本地保存的收敛密钥 K_1 对 C_{K2} 进行解密得 $K_2 = Decrypt(K_1, C_{K2})$，$K_2$ 对 C_{K3} 进行解密得 $K_3 = Decrypt(K_2, C_{K3})$，……，以此类推，递归解密得到所有的收敛密钥$\{K_i\}$，最后将$\{K_i\}$对$\{C_i\}$进行解密恢复出文件 F。

4　安全性分析

4.1　所有权证明

本文方案中，用户必须通过 PoW 协议的验证才能获得文件权限，而不是仅仅通过文件摘要来进行判断。攻击者可能从其他地方获得文件摘要或部分文件内容，企图在 CSP 中获得整个文件。显然，在本文方案中仅凭文件摘无法通过验证，以下讨论的是攻击者拥有部分文件内容的情况。

假设攻击者拥有的部分文件为 m 个数据块，整个文件为 n 个数据块，则所占百分比为 $p = m/n$；那么从 CSP 中随机选取该文件一个数据块，攻击者能够正确响应的概率为 p。将攻击者顺利通过 PoW 验证记为 A 事件，A 事件发生的情况共有两种：1）攻击者响应的数组通过 PoW 协议，记为 R 事件；2）Bloom Filter 发生假阳性误判，误判率为 p_f。则 A 事件发生的概率为：

$$\begin{aligned} P(A) &= P[A\cap(R\cup\bar{R})] \\ &= P(A\mid R)P(R)+P(A\mid\bar{R})P(\bar{R}) \\ &= P(R)+p_f P(\bar{R}) \end{aligned} \quad (2)$$

接下来分析 R 事件发生的概率，即攻击者正确响应数组的概率。CSP 的挑战数组包含 J 个数据块，对攻击者而言，必须根据自己拥有的 m 个数据块去进行响应。由于挑战数组是随机的，可计算得攻击者 m 个数据块中恰好包含 J 个数据块的概率为：

$$P_1=\frac{C_m^J}{C_n^J} \quad (3)$$

当挑战数组中含有攻击者没有的数据块时，攻击者只能猜测数据块内容或者数据块的哈希值，猜测正确的概率记为 P_2，显然 P_2 趋于 0，则：

$$P(R)=P_1+P_2=\frac{C_m^J}{C_n^J} \quad (4)$$

将式（4）代入式（2），得：

$$P(A)=(1-p_f)\frac{C_m^J}{C_n^J}+p_f \quad (5)$$

为方便说明，设整个文件共 n = 200 个数据块，攻击者已拥有其中 m = 100 个，挑战数组 J = 5 个，p_f 为 0.05，则根据方程 4 计算可得 $P(A)$ = 0.078。由此可知，即使攻击者拥有该文件一半的数据，单次响应通过 PoW 协议验证的概率仍然很小。若攻击者不拥有任何该文件数据，则通过验证的概率 $P(A)$ = p_f，正常情况下本文方案 p_f 可忽略不计，对 p_f 的分析和探讨可参见 6.1 节。

4.2 诚实但好奇服务器

由于 CSP 是诚实但好奇的，可能会查看甚至泄漏用户存储在服务器中的数据，本文方案采取收敛加密来抵抗这种威胁。在上传到云服务器之前，先对用户数据进行收敛加密处理，因此存储在云服务器中的是密文数据。由于收敛加密采用的哈希函数是密码学安全的，攻击者在没有收敛密钥的情况下无法解密数据；而收敛密钥也经过加密链机制进行加密，以密文形式存储在服务器中，因此攻击者无法获得收敛密钥。

4.3 抵抗污染攻击

在典型的云存储去重系统中，初始用户上传初始文件 F 和文件摘要 $H(F)$到 CSP，当其他用户利用 $H(F)$请求下载的时候，CSP 将 F 发送给用户；然而，如果攻击者上传初始文件 F'和文件摘要 $H(F)$到 CSP，则其他用户利用 $H(F)$请求下载得到的文件将是 F'而不是 F，于是造成数据污染。在本文方案模型中，若初始上传者发送的密文数据块$\{C_i\}$与标签摘要 h_c 不一致，那么后继上传者将无法通过 PoW 验证，合法用户将无法获得文件权限，也即发生数据污染。因此，为了检查数据块和标签的一致性，本文方案初始用户上传文件时必须上传密文数据块$\{C_i\}$、标签$\{token_i\}$和摘要 h_c，服务器根据用户上传的$\{C_i\}$重新计算标签值和摘要 h'_c，最后比较 h'_c = h_c 是否成立，若成立说明没有数据污染；否则说明存在污染。因此本文方案能够抵抗污染攻击。

表 1 是本文方案与其他基于 Bloom Filter 的 PoW 去重方案的安全性比较。√表示能抵抗，×表示不能抵抗。

表 1　方案的安全性比较

安全性	文献[7]	文献[16]	本文
所有权证明	√	√	√
诚实但好奇服务器	×	√	√
污染攻击	×	×	√

5　性能分析

本文方案和文献[16]同样是结合收敛加密和 Bloom Filter 的 PoW 去重方案，以下是两个方案之间的性能分析和对比。

5.1　Bloom Filter 误判率

本文方案主要考虑 Bloom Filter 的假阳性误判。*active* 状态表示误判率小于上限值，*full* 状态表示误判率达到上限值；记标准 Bloom Filter 的误判率为 f_{SBF}，即公式（1）中的 p_f；记动态 Bloom Filter 的误判率为 f_{DBF}。由 3.1 节知 DBF 中 SBF 的数量为 $s=\lceil n/c\rceil$，若集合中元素不大于单个 SBF 的容量，即 $n\le c$，那么 DBF 实际上就是一个 SBF，误判率 $f_{DBF}=f_{SBF}$；若集合中元素大于单个 SBF 的容量，即 $n>c$，那么 DBF 中有 s - 1 个 SBF 是 *full* 状态和 1 个 SBF 是 *active* 状态。*full* 状态的 SBF 误判率计算方法参考公式（1），*active* 状态的 SBF 误判率计算只需要把元素数量 $n_a=n-c\lfloor n/c\rfloor$ 代入式（1），可得 f_{DBF} 计算公式，如式（6）所示。

文献[16]中 Bloom Filter 采用的是 SBF，因此误判率只有一种情况，则 f_{SBF} 计算公式为式（7）。

$$f_{DBF}=\begin{cases}(1-e^{-\frac{kn}{m}})^k\ , & n\le c\\ 1-(1-(1-e^{-\frac{kc}{m}})^k)^{\lfloor\frac{n}{c}\rfloor}(1-(1-e^{-\frac{k}{m}(n-c\lfloor\frac{n}{c}\rfloor)})^k)\ , & n>c\end{cases} \quad (6)$$

$$f_{SBF}=(1-(1-\frac{1}{m})^{km})^k\approx(1-e^{-\frac{kn}{m}})^k \quad (7)$$

为方便说明，设 $k=4$，$m=2000$，$c=200$，横坐标轴代表集合中元素数量 n，纵坐标轴代表假阳性误判率，得到 DBF 与 SBF 的误判率对比图，如图 4 所示。

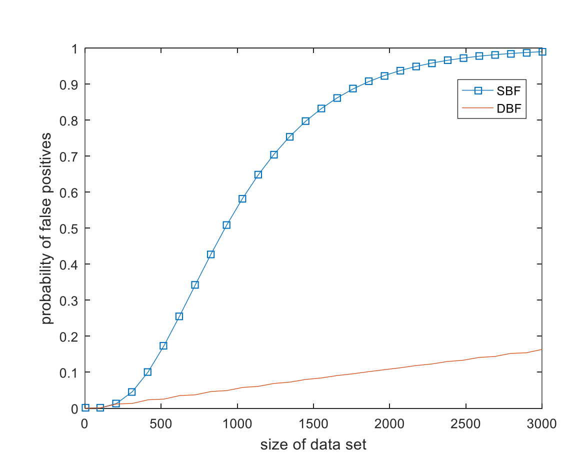


图 4　DBF 与 SBF 的误判率对比图

由图 4 可知，SBF 的误判率随着元素的增多而急剧增加；相比之下，DBF 的误判率增加缓慢，维持在一个可接受的范围内。由于文献[16]采用的是 SBF，而本文方案采用 DBF，因此，相比文献[16]，本文方案的误判率更低。误判率与安全性的分析可参见 5.1 节。

5.2　密钥存储开销

设 n 个用户拥有文件 F 的权限；本文方案采用密钥链机制的思想，对于文件 F，每个用户只需要在本地存储第一个数据块的收敛密钥 K_1，其余数据块的密钥以密文形式$\{C_{Ki}\}$存储在服务器中，并且只需存储一份即可实现跨用户共享。当用户从服务器中下载$\{C_{Ki}\}$后，利用 K_1 对$\{C_{Ki}\}$进行递归解密即可得到全部收敛密钥$\{K_i\}$。由于密钥以密文形式存储在服务器中，因此安全性得到保证。文献[16]方案中，每个用户都用私钥 *userKey* 对收敛密钥进行加密，加密得到的密文各不相同，因此每个用户需要各存储一份密钥密文到服务器；并且每个用户在本地存储一个私钥 *userKey*。

两个方案的密钥存储开销对比如表 2 所示，相比文献[16]，显然本文方案大大降低了密钥存储开销，有效节省服务器存储空间。

表 2　密钥存储开销比较

方案	客户端存储	服务器存储
文献[16]	用户私钥 *userKey*	n 份收敛密钥密文
本文	K_1	1 份收敛密钥密文

注:n 表示拥有该文件权限的用户数

5.3　计算开销

由于一份文件只需上传一次而可能多次运行 PoW 协议，因此主要考虑运行 PoW 协议时的计算开销，也即发生去重时的情况。

a）客户端计算开销。在文献[16]中，计算文件摘要的开销为 T_f，对文件进行分块的计算开销为 T_s，CE 算法的开销为 T_{CE}，计算块标签的开销为 T_{tag}，　*userKey* 对密钥$\{K_i\}$加密的开销为 T_K；在本文中，文件分块的计算开销为 T_s，CE 算法的开销为 T_{CE}，计算块标签的开销为 T_{token}，计算块标签的摘要开销为 T_c。则文献[16]的客户端计算开销为 $T_f+T_s+T_{CE}+T_{tag}+T_K$，本文方案为 $T_c+T_s+T_{CE}+T_{token}$。（2）服务器计算开销。Bloom Filter 利用 k 个哈希函数对 J 个随机数据块验证的计算开销为 T_J，文献[16]中计算用于验证的 L_i 开销为 T_L，本文中计算 PRF 输出结果的开销为 T_{PRF}，。则文献[16]的服务器计算开销为 T_J+T_L，本文方案为 T_J+T_{PRF}。

实验环境：AMD1.8GHz 四核 E2-7110，内存 8GB，Windows 10 系统，编程语言为 Python，收敛密钥产生算法采用 SHA-256，其他哈希函数都采用 SHA-1。

分别对 64MB、128MB、256MB、512MB 和 1GB 的文件进行分块，每个数据块 4KB，每次运行 PoW 协议时挑战数据块 J 的取值是数据块总数的 5%，Bloom Filter 中哈希函数的个数 k 取为 4。计算开销测试结果如图 5 所示。

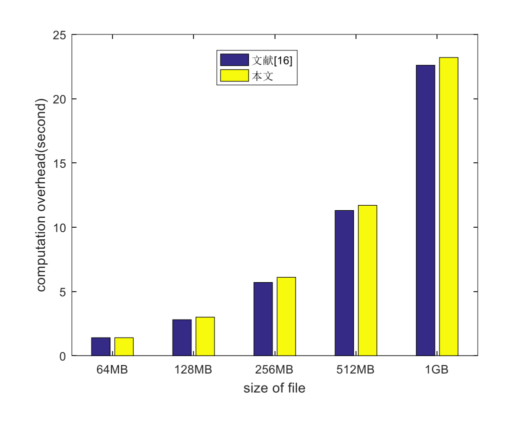


图 5　不同大小文件的计算开销图

通过对比可知，本文方案与文献[16]的计算开销几乎一样。本文为了避免污染攻击（参见 5.3 节），对每个分块都计算其标签值，而文献[16]直接计算整个文件的摘要值，导致本文的计算开销略大一点。

5.4　传输开销

与计算开销的分析一样，主要考虑运行 PoW 协议时的传输开销。挑战阶段，服务器向用户发送包含 J 个数据块索引的挑战数组，用户向服务器返回相关的计算结果，这一部分传输开销本文方案与文献[16]相同。每一次通过 PoW 验证后，文献[16]都需要额外上传收敛密钥的密文；而本文方案无需重复上传收敛密钥，关于密钥的分析可参见 6.2 节。因此，相比文献[16]，本文方案传输开销更小。

例如，对一个 1 GB 的文件进行分块，每个数据块 4 KB，设收敛密钥产生算法为 SHA-256，由于文献[16]采用私钥计算每个块的收敛密钥密文，则计算可得该方案额外传输开销为 8 MB。

6　结束语

针对目前云存储系统数据去重技术中存在的缺陷，本文基于动态 Bloom filter 提出一种改进的 PoW 方案 DBF-Dedup，缓解了标准 Bloom filter 方案中因元素增多而使得误判率快速增加的问题；同时结合了收敛加密算法，不需要可信第三方或复杂的密钥管理，有效抵抗诚实但好奇服务器、污染攻击等安全威胁。安全性与性能分析表明，本文方案在提高安全性的同时，降低了密钥存储开销和传输开销，在安全性和效率之间取得了不错的平衡。

参考文献：

[1] Cisco. Cisco global cloud index (2015-2020) [EB/OL]. (2017) [2018-06-20]. https://www. cisco. com/c/dam/assets/sol/sp/gci/global-cloud-index-infographic. html.

[2] Shin Y, Koo D, Hur J. A survey of secure data deduplication schemes for cloud storage systems [J]. ACM Computing Surveys, 2017, 49 (4): 74.

[3] 熊金波，张媛媛，李凤华，等. 云环境中数据安全去重研究进展 [J]. 通信学报，2016, 37 (11): 169-180. (Xiong Jinbo, Zhang Yuanyuan, Li

Fenghua, *et al.* Research progress on secure data deduplication in cloud [J]. Journal on Communications, 2016, 37 (11): 169-180.)

[4] 陈越，李超零，兰巨龙，等．基于确定//概率性文件拥有证明的机密数据安全去重方案 [J]. 通信学报，2015，36 (9): 1-12. (Chen Yue, Li Chaoling, Lan Julong, *et al.* Secure sensitive data deduplication schemes based on deterministic//probabilistic proof of file ownership [J]. Journal on Communications, 2015, 36 (9): 1-12.)

[5] Halevi S, Harnik D, Pinkas B, *et al.* Proofs of ownership in remote storage systems [C]// Proc of the 18th ACM Conference on Computer and Communications Security. New York: ACM Press, 2011: 491-500.

[6] Pietro R D, Sorniotti A. Boosting efficiency and security in proof of ownership for deduplication [C]// Proc of the 7th ACM Symposium on Information, Computer and Communications Security. New York: ACM Press, 2012: 81-82.

[7] Blasco J, Pietro R D, Orfila A, *et al.* A tunable proof of ownership scheme for deduplication using Bloom filters [C]// Communications and Network Security. Piscataway, NJ: IEEE Press, 2014: 481-489.

[8] Yu Chiamu, Chen Chiyuan, Chao Hanchieh. Proof of ownership in deduplicated cloud storage with mobile device efficiency [J]. Network IEEE, 2015, 29 (2): 51-55.

[9] Xu Jia, Chang Ee-chien, Zhou Jianying. Weak leakage-resilient client-side deduplication of encrypted data in cloud storage [C]// Proc of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. New York: ACM Press, 2013: 195-206.

[10] Bellare M, Keelveedhi S, Ristenpart T. DupLESS: server-aided encryption for deduplicated storage [C]// Proc of the 22nd USENIX Security Symposium. Berkeley, CA: USENIX Association Press, 2013: 179-194.

[11] Puzio P, Molva R, Loureiro S. ClouDedup: secure deduplication with encrypted data for cloud storage [C]// Proc of the 5th IEEE International Conference on Cloud Computing Technology and Science. Piscataway, NJ: IEEE Press, 2013: 363-370.

[12] Stanek J, Sorniotti A, Androulaki E, *et al.* A secure data deduplication scheme for cloud storage [C]// Proc of International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2014: 99-118.

[13] 毕朝国，徐小龙．一种云存储系统中重复数据删除机制 [J]. 计算机应用研究，2014，31 (10): 3052-3055. (Bi Chaoguo, Xu Xiaolong. Data de-duplication mechanism for cloud storage systems [J]. Application Research of Computers, 2014, 31 (10): 3052-3055.)

[14] Miao Meixia, Wang Jianfeng, Li Hui, *et al.* Secure multi-server-aided data deduplication in cloud computing [J]. Pervasive & Mobile Computing, 2015, 24: 129-137.

[15] González-Manzano L, Orfila A. An efficient confidentiality-preserving proof of ownership for deduplication [J]. Journal of Network & Computer Applications, 2015, 50: 49-59.

[16] 刘竹松，杨张杰．基于布隆过滤器所有权证明的高效安全可去重云存储方案 [J]. 计算机应用，2017，37 (3): 766-770. (Liu Zhusong, Yang Zhangjie. Efficient and secure deduplication cloud storage scheme based on proof of ownership by Bloom filter [J]. Journal of Computer Applications, 2017, 37 (3): 766-770.)

[17] 张曙光，咸鹤群，刘红燕，等．云存储中加密数据的自适应重复删除方法 [J]. 计算机应用研究，2018，35 (9): 2772-2776. (Zhang Shuguang, Xian Hequn, Liu Hongyan, *et al.* Adaptive deduplication method for encrypted data in cloud storage [J]. Application Research of Computers, 2018, 35 (9): 2772-2776.)

[18] Bloom B H. Space//time trade-offs in hash coding with allowable errors [J]. Communications of the ACM, 1970, 13 (7): 422-426.

[19] Guo Deke, Wu Jie, Chen Honghui, *et al.* The Dynamic Bloom Filters [J]. IEEE Trans on Knowledge & Data Engineering, 2009, 22 (1): 120-133.

[20] Douceur J R, Adya A, Bolosky W J, *et al.* Reclaiming space from duplicate files in a serverless distributed file system [C]// Proc of the 22nd International Conference on Distributed Computing Systems. Piscataway, NJ: IEEE Press, 2002: 617-624.